Session 123

Advanced Cocoa Topics

Heather Hickman
Cocoa Technology Manager

What We'll Cover

- Build upon the "Cocoa Overview" and "Using Cocoa" sessions
- Provide more in-depth discussions of
  - Drag and drop
  - Table Views
  - Printing
  - Services
  - Run Loops
  - Distributed Objects

AppKit

Mark Piccirelli

Drag and Drop

- An integral part of most user interfaces
- Cocoa makes it easy

Drag and Drop

- Registering drop targets
- Providing drag feedback
- Handling drops
- Initiating drag and drop operations
- New drag and drop features

Registering Drop Targets

- NSViews are the most common drop targets
- Call **-registerForDraggedTypes:** in the view's initializer method
- Call **-unregisterDraggedTypes:** in the view's **-dealloc** method
- Drop targets must implement the NSDraggingDestination protocol

Drag Feedback

The NSDraggingDestination Protocol

- **-draggingEntered:**
  - Draw visual feedback
  - Return drag operation type—none, copy, move, link, delete, generic
  - An NSDraggingInfo object is passed in; gives access to information about the drag

Drag Feedback

The NSDraggingInfo Protocol

- An NSDraggingInfo object is passed to each of your NSDraggingDestination methods
- **-draggingSourceOperationMask**
  - All of the dragging operations that the drag source allows, OR'ed together
- **-draggingPasteboard**
  - Gives access to the data being dragged

Drag Feedback

The NSDraggingDestination Protocol

- **-draggingUpdated:**
  - Redraw feedback if needed; be efficient
  - Autoscrolling is handled by NSScrollView
- **-draggingExited:**
  - Clear visual feedback if needed

Handling Drops

More of the NSDraggingDestination Protocol

- **-prepareForDragOperation:**
  - Return YES if the operation is valid and the pasteboard data is acceptable
- **-performDragOperation:**
  - Fetch data off pasteboard
- **-concludeDragOperation:**
  - Do cleanup, e.g., call **-setNeedsDisplay:**

Initiating Drag and Drop

- **-[NSView dragFile:fromRect:slideBack:event:]** for simple file icon dragging
- **-[NSView dragImage:at:offset:event:pasteboard:source:slideBack:]** for everything else
  - You can use **-[NSBitmapImageRep initWithFocusedViewRect:]** to make an image from part of the view
  - Provide your own implementation of NSDraggingSource for extra control

New Drag and Drop Features

- Two new operations
  - NSDragOperationMove
  - NSDragOperationDelete
- Support for text drag and drop
  - New methods for subclasses and delegates

Table Views

- NSTableView—displays data in a table, and allows the user to edit values and resize and rearrange columns

Name	Role	Types
NSRTFPboardType	Editor	"RTF "
NSRTFDPboardType	Editor	
NSStringPboardType	Editor	"****"
Apple SimpleText document	Editor	TEXT sEXT ttro
Apple HTML document	Viewer	

Table Views

- Providing table view data
- Handling table view events
- Table view cells

Providing Table View Data

The NSTableDataSource Protocol

- **-numberOfRowsInTableView:**
- **-tableView:objectForTableColumn:row:**
  - Standard table view cell is an NSTextFieldCell that can handle NSStrings and NSNumbers
- **-tableView:setObjectValue:forTableColumn:row:**
  - Only required if you want to allow editing of data in the table

Handling Table View Events

NSTableView Delegate Methods

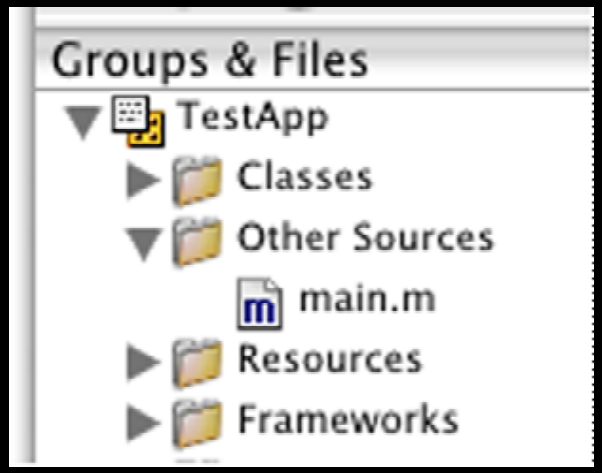
- **-tableView:willDisplayCell:forTableColumn:row:**
  - Set cell attributes for display
- Methods for controlling selection and editing policies
- Methods for receiving notification of selection change, column moving and resizing

Table View Cells

- By default, NSTextFieldCells are used
- Use **-[NSTableColumn setDataCell:]** or override **-[NSTableColumn dataCellForRow:]** to use your own cell class to do custom drawing, special highlighting, or display of custom objects
- Delegate indicates if cell is editable

Outline Views

- NSOutlineView—displays hierarchical data that can be expanded and collapsed, and allows the user to edit values and resize and rearrange columns



Outline Views

- NSOutlineView—subclass of NSTableView
- An NSOutlineViewDataSource provides more information than an NSTableDataSource
  - Whether or not an item is expandable
  - Number of children if so
- Deals with items, not rows
- Persistent object reference replaces row index

Printing

- Know what you get for free
- Simple printing is automatic in many cases
- It's easy to customize printing

Printing

- Simple printing of views
- Page setup
- Print operations
- Pagination

Simple Printing of Views

- **-[NSView print:]** will print out the contents of a view breaking it into pages
- Will use shared NSPrintInfo for page layout
- Presents the Print panel as a dialog box
- Check **+[NSGraphicsContext currentContextDrawingToScreen]** if print output is different from screen

Page Setup

- NSPrintInfo—options from the Page Setup panel (and then some)
- NSPageLayout—presents the Page Setup panel as a dialog or sheet

  -runModalWithPrintInfo:

  -beginSheetWithPrintInfo:modalForWindow: delegate:didEndSelector:contextInfo:

- NSDocument manages nearly all this for you

Print Operations

- NSPrintOperation—specify an NSPrintInfo and an NSView
- Print one of the views already in the window (see the TextEdit sample code)
- Create a view just for printing (see Sketch)
- NSDocument subclasses should override **-printShowingPrintPanel:** and call **-runModalPrintOperation:delegate:didRunSelector:contextInfo:**

Pagination

- For simple cases, set margin and pagination modes in NSPrintInfo
- For more complex situations, override **-[NSView knowsPageRange:]** and return YES
- Override **-[NSView rectForPage:]** and return rectangle in view bounds for the page

File Wrappers

- NSFileWrapper—holds the contents and attributes of a file, or a directory's worth of files, in memory

File Wrappers

- An easy way to manage file attachments (e.g., JPEGs embedded in text)
- An easy way to read file packages (e.g., RTFD documents)
- A file wrapper can contain other file wrappers stored as dictionary entries
- Operations are recursive

File Wrappers

- Get an NSData with the contents of a plain file
- Set and get file attributes and icon
- Check if wrapper contents modified externally and reload
- Create serialized data for transfer via the pasteboard

Services

Doug Davidson

Services

- Provide a way for an application to share its functionality with other applications
- Communication via pasteboards
- Using services is automatic
- Providing a service is simple

Requesting Services

- Add Services menu to your app's main menu—Text views automatically support text and file types
- Can also invoke programmatically
- For custom views, implement:

```
-(id)      validRequestorForSendType:returnType
-(BOOL) writeSelectionToPasteboard:types:
-(BOOL) readSelectionFromPasteboard:
```

- Register new types with NSApplication

Providing a Service

- One method to provide service:

```
serviceName:(NSPasteboard*)pboard
    userData:(NSString *)userData
       error:(NSString **)error
```

- Pasteboard is used for transferring data
- userData lets you provide multiple services through this single entry point
- Set non-nil error message if there was a problem

Handling Service Request

- Get data off pasteboard, change, put new data back on pasteboard:

```
NSString* string = [pboard
stringForType:NSStringPBoardType];

NSString* newString = [self change:string];

[pboard declareTypes:[NSArray arrayWithObject:
NSStringPBoardType] owner:self];

[pboard setStringValue:newString forType:
NSStringPBoardType];
```

Registering Provider

- Register an object as the service provider for your application:

```
Munger* munger = [[Munger alloc] init];
[NSApp setServicesProvider:munger];
```

- One service provider per application
- Register when ready for requests
- Add NSServices information to application settings in Project Builder

Services Info

```
{ NSServices = {
        NSPortName = Munger;
        NSMessage = munge;
        NSSendTypes = (NSStringPBoardType);
        NSReturnTypes =
 (NSStringPBoardType);
        NSMenuItem = {default = “Munge”;}
        NSKeyEquivalent = {default = “M”;}

}}
```

Foundation

Doug Davidson

Run Loops

- Represents a source of inputs
  - User input (mouse, key events)
  - Timers
  - Ports
- Use instead of polling
- One run loop per thread
- A run loop can have multiple modes

Run Loop Modes

- Sources registered for a given mode are called while that mode is active
- Foundation provides a default run loop mode
- AppKit has two additional “common” run loop modes
  - Modal panel
  - Event tracking

Timers

- Synchronous input source
- Single or repeating events
- If the mode is not active at the scheduled time, timer fires when the mode becomes active again
- Use **-[NSTimer invalidate]** to stop a repeating timer

Ports

- Asynchronous input source
- Bi-directional IPC endpoint
- Communicate between tasks or threads
- Allocate port message with sender and receiver ports
- Receiver's delegate receives port message

Distributed Objects

- Lets you send methods to objects transparently across thread, process, and machine boundaries
- Create an NSConnection between two ports
- Local connection uses Mach messages
- NSSocketPort can be used with remote machines to communicate via TCP/IP

Distributed Objects

- Server sets root object for the connection
- Client gets proxies for remote objects, starting with root object
- Method calls are forwarded
- Advertise connection by name with

-[NSConnection registerName:]

Creating a Connection

- Example of interthread messaging:
- In main thread,

```
NSPort* port = [NSPort port];
NSConnection* conn = [[NSConnection alloc]
                    initWithReceivePort:port
sendPort:nil];
[conn setRootObject:self];
[NSThread detachNewThreadSelector:
   @selector(doCalc:) toTarget:[Calc class]
                    withObject:port];
```

Sending Messages

- In Calc class:

```
+ (void)doCalc:(NSPort*)port {
    NSConnection* conn = [[NSConnection alloc]
                          initWithReceivePort:nil
sendPort:port];
    id result = CalculateResult();
    MyServer *server = [conn rootProxy];
    [server reportResult:result];
}
```

NSURL

- NSURL wraps file and network URLs
- Extract and combine components
- Provides simple data fetching for file: and http: schemes
  - One background load at a time
  - Delegate is notified when download starts, finishes, is canceled, or fails

NSURLHandle

- NSURLHandle allows finer control over fetching
- Can handle multiple load requests, caching
- To handle custom schemes, subclass and register

Apple Event Handling

- NSAppleEventManager
  - **-sharedAppleEventManager**
  - **-setEventHandler:andSelector: forEventClass:andEventID:**
  - **-removeEventHandlerForEventClass: andEventID:**
- Selector takes two NSAppleEventDescriptor parameters: incoming event and reply to fill in

Exceptions

- Foundation provides exception mechanism and exception object
- Exceptions have name and reason
- For truly exceptional conditions, usually programming errors
- Can be handled at multiple levels
- If not handled at all, default exception handler logs and exits (can be changed)

Raising Exceptions

- Raise with **+[NSException raise:format:]**
- First parameter is exception name
- Second parameter is human-readable reason, given as format string followed by variable number of arguments
- If you have user info, allocate and **-initWithName:reason:userInfo:**

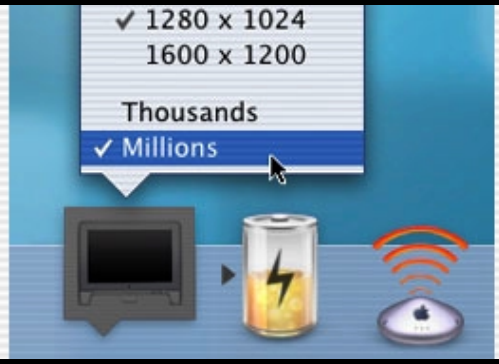
Handling Exceptions

```
NS_DURING
    // ...try operation here
NS_HANDLER
    // ...catch exception here
NS_ENDHANDLER
```

Exception Block

- Within NS_HANDLER…NS_ENDHANDLER block, you may:
  - Fall through
  - Raise the same or a new exception
  - Call **NS_VALUERETURN()**
  - Call **NS_VOIDRETURN**
- Block contains local variable, localException, that can be saved and/or re-raised

New Features



Chuck Pisula

XML

- Cocoa now uses XML-based file format for storing property lists
- Supports some predefined types
  - Array, data, date, dictionary, real, integer, string, boolean
- To use other object types convert to an NSString or NSData

HFS File Types

- Cocoa now loads files based on Mac OS 9 file types as well as extensions
- File types are passed into existing extension-based API, using:

```
NSString *NSFileTypeForHFSTypeCode(OSType)
OSType NSHFSTypeCodeFromFileType(NSString *)
```

- To get the file type of an existing file

```
NSString *NSHFSTypeOfFile(NSString *)
```

NSWorkspace

- It's back!...mostly
- Most functionality is restored with a few exceptions
- New methods:

  -openURL:

  -isFilePackageAtPath:

  -mountedLocalVolumePaths

Custom Window Shapes

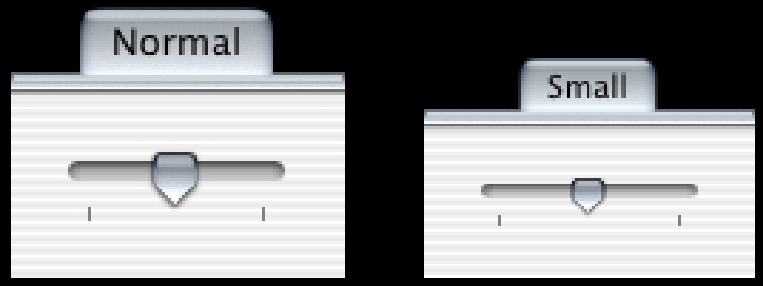
- Can set overall transparency of a window:

  -[NSWindow setAlphaValue:]

- Can make shaped windows via alpha content:
  - **[window setOpaque:NO];**
  - Fill content with **[NSColor clearColor]**
  - Draw where you want opaque area
- Can turn shadow on and off:

  -[NSWindow setHasShadow:]

Small Controls



- A number of controls now have small variants
  - NSButton (checkbox, radio button, pushbutton)
  - NSProgressIndicator, NSPopUpButton
  - NSSlider, NSScroller
  - NSTabView
- Choose small controls…
  - In Interface Builder, choose “Small” option
  - Programmatically via

```
[cntrl setControlSize:NSSmallControlSize]
```

NSStatusBar

- It's back! No new API
- NSStatusItem API is pretty simple
  - Set target, action, title, image, menu, tooltip
  - Items can be custom views: **-setView:**
- Some guidelines to follow
  - Use sparingly and keep items small
  - Consider changing dock icon to show status

NSMovieView

- It's back!
- Encapsulates simple functionality for playing movies—Access Movie directly to do more complex operations
- Movie is now stored in an NSMovie
- MovieController can be accessed from NSMovieView

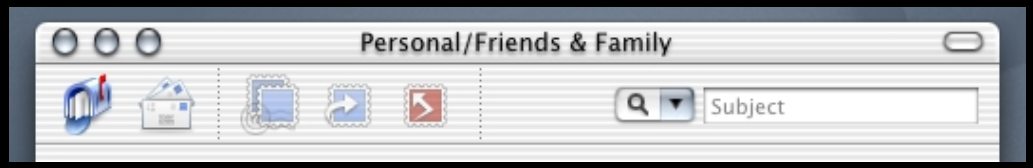
NSMovie

- Wraps QuickTime Movie
- Not an Objective-C cover for QuickTime APIs
- Initialize with QuickTime Movie, URL, or from pasteboard
- Will dispose of movie when deallocated
- Currently, URLs must be file-based—Use **NewMovieFromDataRef()** to create Movie from streaming URL

NSOpenGLView

- New methods update and reshape
  - Reshape is called when view geometry changes (e.g., visible bounds)
  - Call **[super update]** if you override update
- If you use a full screen context, you should provide a cover window to intercept mouse events. Use **kCGOverlayWindow** level to cover dock and menu bar

NSToolbar



- NSToolbar represents the whole toolbar
  - Synchronizes toolbars with the same name
  - Runs the customization sheet
- Each item in the toolbar is an NSToolbarItem
  - Can have an image, label, menu, target, action
  - Can provide a custom view
  - Some standard items: Separators, colors, . . .

NSStepper

- It's new!
- Small up/down arrows
- Commonly used for date and time entry
- Set min, max, increment, wrap, autorepeat

NSImage

- Two new methods

  -drawAtPoint:fromRect:operation:fraction:
  -drawInRect:fromRect:operation:fraction:

- Always respect current transform (scale, rotate) unlike existing compositeToPoint:
- Can specify both composite operation and alpha
- Can specify portion of source image

New Graphics Features

- NSGraphicsContext now gives you finer control
- Anti-aliasing

  -setShouldAntialias:, -shouldAntialias
- Image interpolation

  -setImageInterpolation: / -imageInterpolation

  NSImageInterpolation (none, low, high, default)

New Graphics Features

- NSBitmapImageRep can now use ColorSync data
- Stored in property dictionary with **NSImageColorSyncProfileData** as the key

Versioning

- Test your application against older versions of Mac OS X
- Check **NSAppKitVersionNumber** (577.0 in 10.0.0)
- In 10.0, you will need to declare it yourself:

```
extern double NSAppKitVersionNumber
```

- Use **respondsToSelector:** to check for new methods
- Check release notes for changes in functionality

Documentation

- Release notes
- Overview documentation
- Programming topics covered in-depth
- Class references
- Example code

Documentation

- Documentation in
  - /Developer/Documentation/Cocoa
  - http://developer.apple.com/techpubs/ under Mac OS X
- Example code in
  - /Developer/Examples/Foundation
  - /Developer/Examples/AppKit
  - /Developer/Examples/Java/AppKit

Roadmap

102 Cocoa Overview: Introduction to Cocoa	Hall 2 **Mon., 5:00pm**
122 Using Cocoa: Classes and features in depth	Hall 2 **Thurs., 3:30pm**
FF112 Feedback Forum: Cocoa: Have your say	Room A2 **Fri., 5:00pm**

Who to Contact

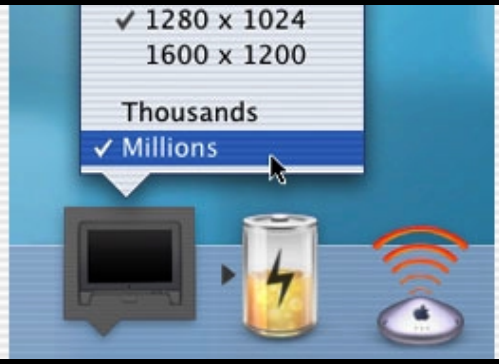
Heather Hickman
Cocoa Technology Manager
Apple Worldwide Developer Relations
hhickman@apple.com

Cocoa Feedback
cocoa-feedback@apple.com

Cocoa Development Mailing List
Subscribe at
www.lists.apple.com/mailman/listinfo/cocoa-dev

http://developer.apple.com/events/urls.html

Q&A



Heather Hickman
Cocoa Technology Manager
hhickman@apple.com

http://developer.apple.com/events/urls.html

WWDC

Worldwide Developers Conference 2001

WWDC

Worldwide Developers Conference 2001

WWDC

Worldwide Developers Conference 2001